Overview of Algorithms for Swarm Intelligence

Shu-Chuan Chu[1], Hsiang-Cheh Huang[2], John F. Roddick[1], and Jeng-Shyang Pan[3]

[1] School of Computer Science, Engineering and Mathematics,
Flinders University of South Australia, Australia
[2] National University of Kaohsiung, 700 University Road, Kaohsiung 811, Taiwan, R.O.C.
[3] National Kaohsiung University of Applied Sciences, 415 Chien-Kung Road,
Kaohsiung 807, Taiwan, R.O.C.

Abstract. Swarm intelligence (SI) is based on collective behavior of self-organized systems. Typical swarm intelligence schemes include Particle Swarm Optimization (PSO), Ant Colony System (ACS), Stochastic Diffusion Search (SDS), Bacteria Foraging (BF), the Artificial Bee Colony (ABC), and so on. Besides the applications to conventional optimization problems, SI can be used in controlling robots and unmanned vehicles, predicting social behaviors, enhancing the telecommunication and computer networks, etc. Indeed, the use of swarm optimization can be applied to a variety of fields in engineering and social sciences. In this paper, we review some popular algorithms in the field of swarm intelligence for problems of optimization. The overview and experiments of PSO, ACS, and ABC are given. Enhanced versions of these are also introduced. In addition, some comparisons are made between these algorithms.

Keywords. Swarm intelligence (SI), Particle Swarm Optimization (PSO), Ant Colony System (ACS), Artificial Bee Colony (ABC).

1 Introduction

People learn a lot from Mother Nature. Applying the analogy to biological systems with lots of individuals, or swarms, we are able to handle the challenges in the algorithm and application with optimization techniques. In this paper, we focus on the overview of several popular swarm intelligence algorithms, pointing out their concepts, and proposing some enhancements of the algorithms with the results of our research group.

Swarm intelligence, according to [1], is the emergent collective intelligence of groups of simple agents. With swarm intelligence, the developed algorithms need to be flexible to internal and external changes, to be robust when some individuals fail, to be decentralized and self-organized [2]. In the rest of the paper, we will address several popular algorithms based on these concepts, including Particle Swarm Optimization (PSO), Ant Colony System (ACS), and Artificial Bee Colony (ABC) algorithms in Sec. 2, and we present the improvements of these algorithms based on our existing works in Sec. 3. Selected simulation results and comparisons are also provided in Sec. 4. Finally, we conclude this paper in Sec. 5.

P. Jędrzejowicz et al. (Eds.): ICCCI 2011, Part I, LNCS 6922, pp. 28–41, 2011.

2 Swarm Intelligence Algorithms

In this section, we introduce the concept and implementation of several popular algorithm for swarm intelligence optimization, including particle swarm optimization (PSO), ant colony system (ACS), and Artificial Bees Colony (ABC) algorithms.

2.1 Particle Swarm Optimization (PSO)

Particle swarm optimization (PSO) was first introduced by Kennedy and Eberhart [3,4]. It is a relatively new stochastic optimization technique that can simulate the swarm behavior of birds flocking. In PSO, an individual in the swarm, called a particle, represents a potential solution. Each particle has a fitness value and a velocity, and it learns the experiences of the swarm to search for the global optima [5]. Traditional PSO can be depicted in Fig. 1. They include (1) particle initialization, (2) velocity updating, (3) particle position updating, (4) memory updating, and (5) termination checking. These steps are described as follows.

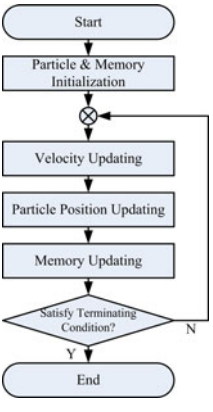


Fig. 1. Procedures for particle swarm optimization

(1) *Initialization*. We first decide how many particles used to solve the problem. Every particle has its own position, velocity and best solution. If we use *M* particles, their best solutions, and their velocities can be represented as:

$$\mathbf{X} = \{x_0, x_1, \cdots, x_{M-1}\}, \tag{1}$$

$$\mathbf{B} = \{b_0, b_1, \cdots, b_{M-1}\}, \tag{2}$$

$$\mathbf{V} = \{v_0, v_1, \cdots, v_{M-1}\}. \tag{3}$$

(2) *Velocity updating*. This step is shown in Eq. (4), where c_1 and c_2 are constants, r_1 and r_2 are random variables in the range from 0 to 1, $b_i(t)$ is the best solution of the *i*-th particle for the iteration number up to the *t*-th iteration and the $G(t)$ is the best solution of all particles:

$$v_i(t+1) = v_i(t) + c_1 \cdot r_1 \cdot (b_i(t) - x_i(t)) + c_2 \cdot r_2 \cdot (G(t) - x_i(t)). \tag{4}$$

To prevent the velocity from becoming too large, we set a maximum value to limit the range of velocity as $-V_{MAX} \le V \le V_{MAX}$.

(3) *Position updating*, which is processed by Eq. (5):

$$x_i(t+1) = x_i(t) + v_i(t), \quad i = 0, 1, \cdots, M-1. \tag{5}$$

(4) *Memory updating*. If we find a better solution than $G(t)$ in $G(t+1)$, $G(t)$ will be replaced by $G(t+1)$. Otherwise, there will be no change for $G(t)$.

(5) These recursive steps continue unless we reach the termination condition.

With the descriptions above, we can observe that the solution in PSO can be influenced by both the global and the local characteristics in the training procedure.

2.2 Any Colony Systems

Inspired by the food-seeking behavior of real ants, the ant system [6,7] is a cooperative population-based search algorithm. As each ant constructs a route from nest to food by stochastically following the quantities of pheromone level, the intensity of laying pheromone would bias the path-choosing, decision-making of subsequent ants.

The operation of ant system can be illustrated by the classical traveling salesman problem (TSP). The TSP seeks for a round route covering all cities with minimal total distance. Suppose there are *n* cities and *m* ants. The entire algorithm starts with initial pheromone intensity set to s_0 on all edges. In every subsequent ant system cycle, or the episode, each ant begins its tour from a randomly selected starting city and is required to visit every city once and only once. The experience gained in this phase is then used to update the pheromone intensity on all edges.

The algorithm of the ant system for the TSP is depicted as follows [7,8]:

(1) Randomly select the initial city for each ant. The initial pheromone level between any two cities is set to be a small positive constant. Set the cycle counter to be 0.

(2) Calculate the transition probability from city *r* to city *s* for the *k*-th ant as

$$P_k(r,s) = \begin{cases} \dfrac{[\tau(r,s)] \cdot [\eta(r,s)]^{\beta}}{\sum_{u \in J_k(r)} [\tau(r,u)] \cdot [\eta(r,u)]^{\beta}}, & \text{if } s \in J_k(r), \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where *r* is the current city, *s* is the next city, $\tau(r,s)$ is the pheromone level between cities *r* and *s*, $\eta(r,s) = \delta(r,s)^{-1}$ is the inverse of the distance $\delta(r,s)$ between cities *r* and *s*, $J_k(r)$ is the set of cities that remain to be visited by the *k*-th ant positioned on city *r*, and *b* is a parameter determining the relative importance of pheromone level versus distance. Select the next visited city *s* for the *k*-th ant with the probability $P_k(r,s)$. Repeat Step (2) for each ant until the ants have toured all cities.

(3) Update the pheromone level between cities as

$$\tau(r,s) \leftarrow (1-\alpha)\cdot\tau(r,s)+\sum_{k=1}^{m}\Delta\tau_k(r,s), \tag{7}$$

$$\Delta\tau_k(r,s)=\begin{cases}\dfrac{1}{L_k}, & \text{if } (r,s)\in \text{route done by ant } k,\\ 0, & \text{otherwise,}\end{cases} \tag{8}$$

where $\Delta\tau_k(r,s)$ is the pheromone level laid down between cities r and s by the k-th ant, L_k is the length of the route visited by the k-th ant, m is the number of ants and $0<\alpha<1$ is a pheromone decay parameter.

(4) Increment cycle counter. Move ants to originally selected cities and continue Steps (2)–(4) until the behavior stagnates or the maximum number of cycles has reached. A stagnation is indicated when all ants take the same route.

From Eq. (6) it is clear ant system (AS) needs a high level of computation to find the next visited city for each ant. In order to improve the search efficiency and lower computational complexity, the ant colony system (ACS) was proposed [8,9]. ACS is based on AS but updates the pheromone level before moving to the next city (local updating rule) and updating the pheromone level for the shortest route only after completing the route for each ant (global updating rule) as

$$\tau(r,s) \leftarrow (1-\alpha)\cdot\tau(r,s)+\alpha\cdot\Delta\tau(r,s), \tag{9}$$

$$\Delta\tau(r,s)=\begin{cases}\dfrac{1}{L_{gb}}, & \text{if } (r,s)\in \text{global best route,}\\ 0, & \text{otherwise,}\end{cases} \tag{10}$$

where L_{gb} is the length of the shortest route and α is a pheromone decay parameter.

2.3 Artificial Bee Colony (ABC) Optimization

Karaboga proposed Artificial Bee Colony (ABC) [10] in 2005 based on inspecting the behaviors of real bees on finding nectar and sharing the information of food sources to the bees in the nest. Three kinds of bees are defined as the artificial agents. Every kind of bee plays different and important roles in the optimization process. The artificial agents are called the employed bee, the onlooker, and the scout with distinct responsibilities. The employed bee stays on a food source, which represents a spot in the solution space, and provides the coordinate for the onlookers in the hive for reference. The onlooker bee receives the locations of food sources and selects one of the food sources to gather the nectar. The scout bee moves in the solution space to discover new food sources. The process of the ABC optimization is described as follows:

(1) *Initialization*. Spray n_e percentage of the populations into the solution space randomly, and then calculate their fitness values, called the nectar amounts, where n_e represents the ratio of employed bees to the total population. Once these populations are positioned into the solution space, they are called the

employed bees. Evaluate the fitness of the employed bees and take the fitness to be their amount of nectar.

(2) Move the onlookers. We calculate the probability of selecting a food source by Eq. (9) first, where θ_i denotes the position of the *i*-th employed bee, $F(\bullet)$ is the fitness function, *S* represents the number of employed bees, and P_i is the probability of selecting the *i*-th employed bee. Then we select a food source to move to by roulette wheel selection for every onlooker bee and determine the nectar amounts. The onlookers are moved by Eq. (10), where x_i denotes the position of the *i*-th onlooker bee, t denotes the iteration number, θ_k is the randomly chosen employed bee, *j* represents the dimension of the solution, and $\phi(\bullet)$ produces a series of random variable in the range $[-1, 1]$.

$$P_i = \frac{F(\theta_i)}{\sum_{k=1}^{S} F(\theta_k)}, \tag{11}$$

$$x_{ij}(t+1) = \theta_{ij}(t) + \phi(\theta_{ij}(t) - \theta_{kj}(t)). \tag{12}$$

(3) Update the best food source found so far. We record the best fitness value and the position, which are found by the bees.

(4) Move the scouts. If the fitness values of the employed bees are not improved by a consecutive number of iterations, called "Limit," those food sources are abandoned, and these employed bees become the scouts. The scouts are moved by Eq. (11), where *r* is a random number and $r \in [0, 1]$.

$$\theta_{ij} = \theta_{j\min} + r \cdot (\theta_{j\max} - \theta_{j\min}) \tag{13}$$

(5) Termination checking. If the amount of the iterations satisfies the termination condition, we terminate the program and output the results; otherwise, go back to Step (2).

2.4 Relating Algorithms with SI

Due to the limited space in this paper, readers who are interested in relating algorithms are suggested to refer to the followings: Bacteria Foraging (BF) [11], Cat Swarm Optimization (CSO) [14,13], and Stochastic Diffusion Search (SDS) [14].

3 The Enhanced Swarm Intelligence Algorithms

3.1 Parallel PSO

A parallel computer consists of a large number of processing elements which can be dedicated to solving a single problem at a time. Pipeline processing and data parallelism are two popular parallel processing methods. The function of the pipeline processing is to separate the problem into a cascade of tasks where each task is executed by an individual processor, while data parallelism involves distributing the

data to be processed amongst all processors which then executes the same procedure on each subset of the data. And this is the motivation for the parallel PSO.

Computation time of PSO in Sec. 2.1 can be reduced with the parallel structure. Parallel processing aims at producing the same results achievable using multiple processors with the goal of reducing the run time. We also have the same steps in describing parallel PSO (PPSO) in Sec. 2.1. But in Step (1), users must define how many groups need be in the process because it can be designed to be 2^n sets. To decide the number of groups, it should be carefully refer to the size of the problem to solve. In general, we use four groups to evolve, but we use eight groups only if the problem is quite large. After several iterations, each group exchanges information in accordance with an explicit rule chosen by users.

The parallel particle swarm optimization (PPSO) method [15] gives every group of particles the chance to have the global best and the local best solutions of other groups. It increases the chance of particles to find a better solution and to leap the local optimal solutions.

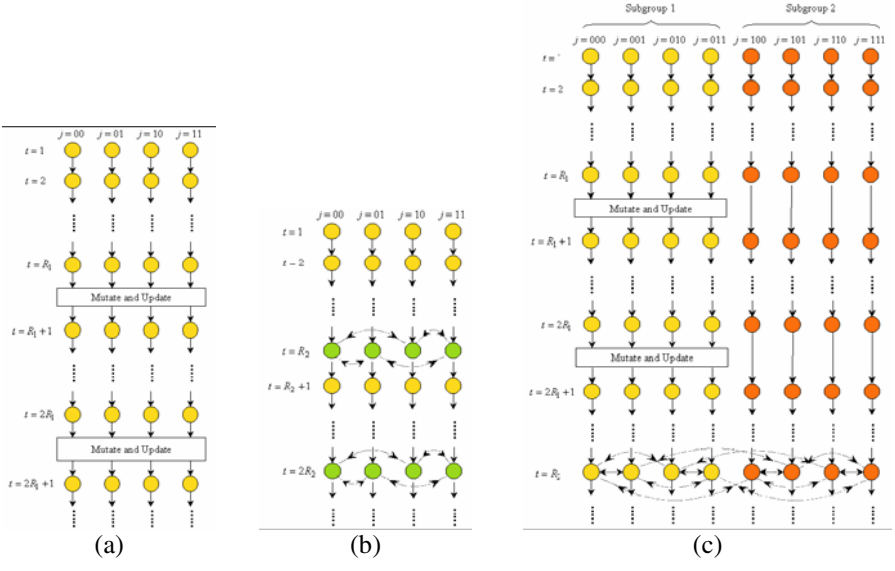


Fig. 2. Communication strategies for parallel PSO. (a) Strategy #1, with loose correlation. (b) Strategy #2, with strong correlation. (c) Strategy #3, with general purpose usage.

Three communication strategies for PPSO are presented. The first one is to deal with the loosely correlated parameters in functions. When all the particles evolved a period of time (Here we described as R_1 iterations), communication strategy 1 would exchange the local best solution from one group to others. The way of communication strategy 1 is shown in Fig. 2(a).

The second communication strategy is the self-adjustment in each group due to the strong correlations of parameters. In each group, the best particle would migrate to its neighborhood to supplant some particles by replacing a deteriorated solution with the

group's best one. For the ease of implementation, the number of groups for the parallel structure is set to a power of two. Thus, neighborhoods are defined as one bit difference in the binary format. The second communication strategy would be applied every R_2 iterations for replacing the poorly performing particles. The way of communication strategy 2 is shown in Fig. 2(b).

The third communication strategy of PPSO is the combination of communication strategies 1 and 2. The particles must be separate into at least four groups. In communication strategy 3, all particles are separated into two subgroups. Under the subgroups, communication strategy 1 and 2 are imitated. Thus, communication strategy 3 is a general communication strategy for PPSO. The process is shown in Fig. 2(c).

Based on the observation, if the parameters are loosely correlated or independent, communication strategy 1 will obtain good results quite quickly. On the contrary, communication strategy 2 obtains higher performance if the parameters are tightly correlated. However, these communication strategies may perform poorly if they have been applied in the wrong situation. Consequently, when the correlation of parameters is unknown, communication strategy 3 is the better choice to apply.

3.2 Parallel ACS

Similar to PPSO, we apply the idea of data parallelism to ant colony system (ACS) in Sec. 2.2 to reduce running time and obtain a better solution. The parallel ant colony system (PACS) based on traveling salesman problem is described as follows:

(1) *Initialization*. Generate N_j artificial ants for the *j*-th group, $j = 0, 1, \cdots, G-1$, and *G* is the number of groups. Randomly select an initial city for each ant. The initial pheromone level between any two cities is set to be a small positive constant τ_0. Set the cycle counter to be 0.

(2) *Movement*. Calculate the next visited city *s* for the *i*-th ant in the *j*-th group according to

$$s = \arg\max_{u \in J_{i,j}(r)} \left[\tau_j(r,u)\right] \cdot \left[\eta(r,u)\right]^{\beta} \quad \text{if } q \le q_0 \text{ (exploitation)} \tag{14}$$

and visit city *s* with $P_{i,j}(r,s)$ if $q > q_0$ (biased exploration);

$$P_{i,j}(r,s) = \begin{cases} \dfrac{\left[\tau_j(r,s)\right] \cdot \left[\eta(r,s)\right]^{\beta}}{\sum_{u \in J_k(r)} \left[\tau_j(r,u)\right] \cdot \left[\eta(r,u)\right]^{\beta}}, & \text{if } s \in J_{i,j}(r), \\ 0, & \text{otherwise,} \end{cases} \tag{15}$$

where $P_{i,j}(r,s)$ is the transition probability from city *r* to city *s* for the *i*-th ant in the *j*-th group. $\tau_j(r,s)$ is the pheromone level between city *r* to city *s* in the *j*-th group. $\eta(r,s) = \delta(r,s)^{-1}$ the inverse of the distance $\delta(r,s)$ between city *r* and city *s*. $J_{i,j}(r)$ is the set of cities that remain unvisited by the *i*-th ant in the *j*-th group and β is a parameter which determines the relative importance of pheromone level versus distance. *q* is a random number between 0 and 1 and q_0 is a constant between 0 and 1.

(3) *Local pheromone level updating rule*. Update the pheromone level between cities for each group as

$$\tau_j(r, s) \leftarrow (1-\alpha)\cdot\tau_j(r, s) + \rho\cdot\Delta\tau(r, s), \tag{16}$$

$$\Delta\tau(r, s) = \tau_0 = \frac{1}{n\cdot L_{nn}}, \tag{17}$$

where $\tau_j(r, s)$ is the pheromone level between cities r and s for the ants in the j-th group, L_{nn} is an approximate distance of the route between all cities using the nearest neighbor heuristic, n is the number of cities and $0 < \rho < 1$ is a pheromone decay parameter. Continue Steps 2 and 3 until each ant in each group completes the route.

(4) *Evaluation*. Calculate the total length of the route for each ant in each group.

(5) *Global pheromone level updating rule*. Update the pheromone level between cities for each group as

$$\tau_j(r, s) \leftarrow (1-\alpha)\cdot\tau_j(r, s) + \rho\cdot\Delta\tau_j(r, s), \tag{18}$$

$$\Delta\tau_j(r, s) = \begin{cases} \dfrac{1}{L_j}, & \text{if } (r, s) \in \text{best route of } j\text{-th group}, \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

where L_j is the shortest length for the ants in the j-th group and α is a pheromone decay parameter.

(6) *Updating from communication*. Seven strategies are proposed as follows:

I. As shown in Fig. 3(a), update the pheromone level between cities for each group for every R_1 cycles as

$$\tau_j(r, s) \leftarrow \tau_j(r, s) + \lambda\cdot\Delta\tau_{\text{best}}(r, s), \tag{20}$$

$$\Delta\tau_{\text{best}}(r, s) = \begin{cases} \dfrac{1}{L_{\text{gb}}}, & \text{if } (r, s) \in \text{best route of all groups}, \\ 0, & \text{otherwise}, \end{cases} \tag{21}$$

where k is a pheromone decay parameter and L_{gb} is the length of the best route of all groups, i.e. $L_{\text{gb}} \le L_j,\ j = 0, 1, \cdots, G-1$.

II. As shown in Fig. 3(b), update the pheromone level between cities for each group for every R_2 cycles as

$$\tau_j(r, s) \leftarrow \tau_j(r, s) + \lambda\cdot\Delta\tau_{\text{ng}}(r, s), \tag{22}$$

$$\Delta\tau_{\text{ng}}(r, s) = \begin{cases} \dfrac{1}{L_{\text{ng}}}, & \text{if } (r, s) \in \text{best route of neighbor groups}, \\ 0, & \text{otherwise}, \end{cases} \tag{23}$$

where neighbor is defined as being the group whose binary representation of the group number j differs by the least significant bit. λ is a pheromone decay parameter and L_{ng} is the length of the shortest route

in the neighbor group.

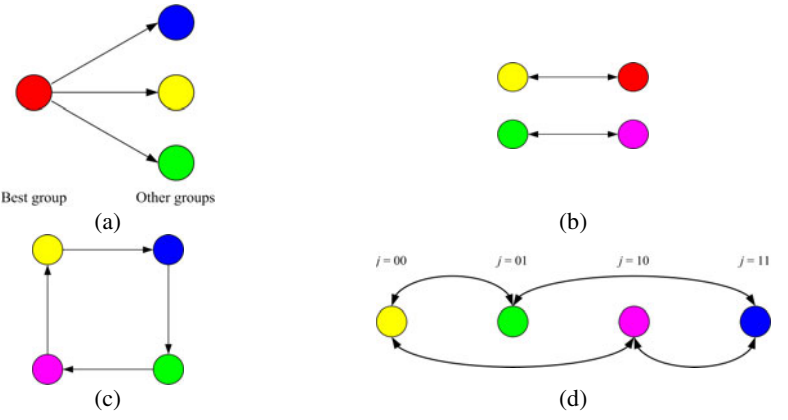


Fig. 3. Updating schemes for communication. (a) Update the pheromone level between cities for each group for every R_1 cycles. (b) Update the pheromone level between cities for each group for every R_2 cycles. (c) Update the pheromone between cities for each group for every R_3 cycles. (d) Update he pheromone between cities for each group for every R_4 cycles.

III. As shown in Fig. 3(c), update the pheromone between cities for each group for every R_3 cycles as

$$\tau_j(r, s) \leftarrow \tau_j(r, s) + \lambda \cdot \Delta\tau_{ng}(r, s), \tag{24}$$

$$\Delta\tau_{ng}(r, s) = \begin{cases} \dfrac{1}{L_{ng}}, & \text{if } (r, s) \in \text{best route of neighbor groups,} \\ 0, & \text{otherwise,} \end{cases} \tag{25}$$

where neighbor is defined as being the group arranged as the ring structure. λ is a pheromone decay parameter and L_{ng} is the length of the shortest route in the neighbor group.

IV. As shown in Fig. 3(d), update the pheromone between cities for each group for every R_4 cycles as

$$\tau_j(r, s) \leftarrow \tau_j(r, s) + \lambda \cdot \Delta\tau_{ng}(r, s), \tag{26}$$

$$\Delta\tau_{ng}(r, s) = \begin{cases} \dfrac{1}{L_{ng}}, & \text{if } (r, s) \in \text{best route of neighbor groups,} \\ 0, & \text{otherwise,} \end{cases} \tag{27}$$

where neighbor is defined as being those groups where the binary representation of the group number *j* differs by one bit. λ is a pheromone decay parameter and L_{ng} is the length of the shortest route in the neighbor group.

V. Update the pheromone between cities for each group using both Strategies 1 and 2.
VI. Update the pheromone between cities for each group using both Strategies 1 and 3.
VII. Update the pheromone between cities for each group using both Strategies 1 and 4.

(7) Termination. Increment the cycle counter. Move the ants to the originally selected cities and continue Steps 2–6 until the stagnation or a present maximum number of cycles has reached, where a stagnation indicated by all ants taking the same route.

With the parallel formulation for the ant colony system, seven communication strategies between groups which can be employed to update the pheromone levels are presented. Details of the parallel ant colony system (PACS) can be observed in [9] with better results for the traveling salesman problem.

3.3 ABC with Communication Strategy

Regarding to Sec. 2.3, to use forward-communication strategy in ABC optimization, the artificial agents in ABC are split into G subpopulations. Each subpopulation evolves by ABC optimization independently, or the subpopulation has its own near best solution, and the artificial agents do not know there are other subpopulations in the solution space. The total iteration contains R times of forward-communications, where $R = \{R_1, 2R_1, 3R_1, ...\}$. The diagram of the parallelized structure of ABC optimization is given in Fig. 4; the diagram of $k = 1$ forward-communication strategy is given in Fig. 5.

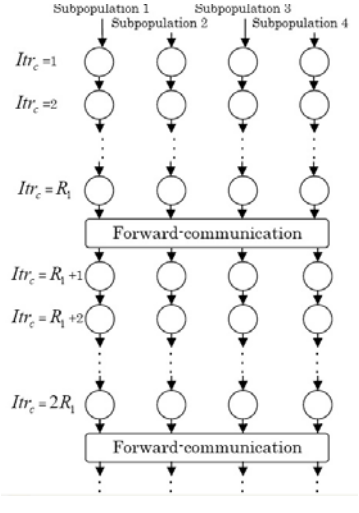


Fig. 4. The diagram of ABC optimization

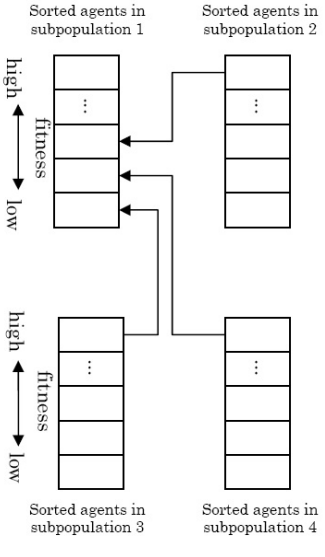


Fig. 5. The diagram of the forward-communication.

When the forward-communication process is executed, the predefined first subpopulation receives $(G-1)\times k$ near best solutions from other subpopulations, where k is the number of agents picked to replace the worst agents in the first subpopulation. The first subpopulation adopts the received near best solutions and wipes out the same number of the worst agents in its population. The subpopulations pop k near best solutions ahead to the first subpopulation in the forward-communication process, and then the subpopulations return to the status with no interaction again. The process of ABC with forward-communication strategy, called ABC-FC, is described as follows:

(1) *Initialization*. Generate the artificial agents and divide them into G subpopulations. Each subpopulation is initialized by ABC independently. Defined the iteration set R for executing the forward-communication.
(2) *Evolvement of the subpopulations*. Evolve the subpopulations independently by ABC optimization.
(3) *Forward-communication*. Let Itr_c denotes the current iteration. If $Itr_c \cap R \neq \phi$, replace the worst $(G-1)\times k$ agents in the first subpopulation by the near best solutions collected from other subpopulations. Otherwise, skip this step.
(4) *Termination checking*. If the termination condition is satisfied, output the result and terminate the program. Otherwise go back to Step (2).

With the proposed ABC with forward-communication strategy, it splits the artificial agents into independent subpopulations and provides the forward-communication to the predefined first subpopulation. Better results can be observed in [16].

4 Some Comparisons

Three well-known test functions, namely, Schwefel's function, Rastrign function, and Griewank function, are emplyed to test the accuracy and the convergence of ABC, PSO, and ABC with the forward-communication strategy. The test functions are listed in Eq. (28) to Eq. (30) as follows.

$$f_1(X) = \sum_{i}^{n} (\sum_{j}^{i} x_j)^2 \tag{28}$$

$$f_2(X) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10) \tag{29}$$

$$f_3(X) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{30}$$

The goal of the optimization is to minimize the outcome. The initial ranges for the test functions are listed in Table 1. For all algorithms, the total population size is 100 and the dimension of the solution space is 100. Each algorithm is executed with 5000 iterations and is repeated with 30 runs. The final result is obtained by taking the average of the outcomes from all runs. The weighting factor of PSO is set to be linearly decreased from 0.9 to 0.4, $c_1 = c_2 = 2$, $r_1, r_2 \in [0,1]$ are random variables in Eq. (4).

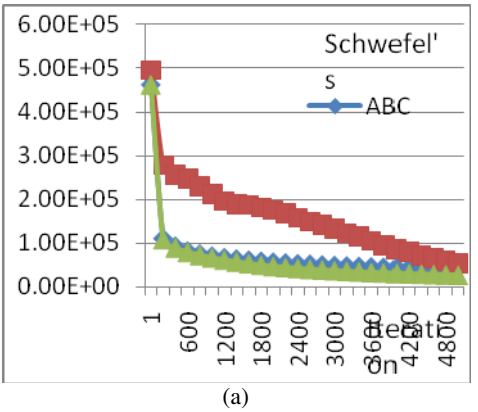
Table 1. Initial ranges for the test functions

Function	Initial range
f_1	$-65.536 \le x_i \le 65.536$
f_2	$-5.12 \le x_i \le 5.12$
f_3	$-600 \le x_i \le 600$

Table 2. The time consumption for obtaining the near best solutions

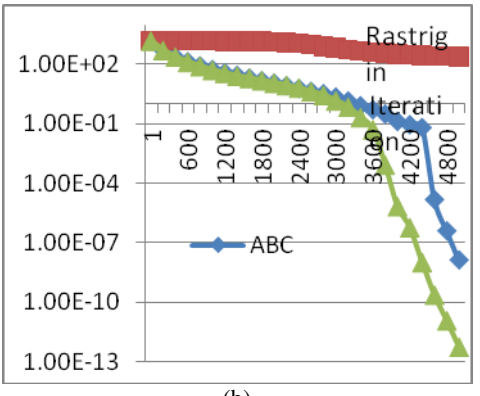
	Function	ABC	PSO	ABC-FC
f_1	Fitness value	4.267×10^{4}	5.389×10^{4}	2.800×10^{4}
	Time cost (Sec.)	45.08	32.20	44.57
f_2	Fitness value	1.286×10^{-8}	1.286×10^{-8}	5.125×10^{-13}
	Time cost (Sec.)	8.448	1.215	8.466
f_3	Fitness value	2.319×10^{-15}	5.877×10^{-3}	3.593×10^{-16}
	Time cost (Sec.)	15.99	15.48	14.28

For ABC with forward-communication strategy, $G=4$, $R_1=100$, and $k=5$ are used. The experimental results are presented in Fig. 6 and the time cost for finding the near best solutions are listed in Table 2. Comparing to the results obtained by PSO and ABC, the accuracy of ABC-FC improves 83% and 73% comparatively. The time cost of finding the near best solutions is reduced about 4% than ABC.

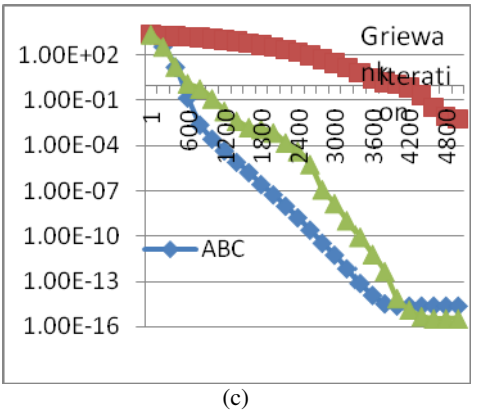


(a)

Fig. 6. Experimental results of the test functions. (a) Schwefel's function. (b) Rastrigin function. (c) Griewank function



(b)



(c)

Fig. 6. (*continued*)

5 Conclusions

In this paper, we have introduced an overview of several popular algorithms for swarm optimization. Enhancements of corresponding algorithms are also presented, and they are verified by simulations with the traveling salesman problem and popularly employed test functions. We observe that there might be analogies and relationships in distributed computing and social insects. Therefore, developments of new algorithms to a variety of applications of swarm intelligence must be interesting research topics for further studies.

References

1. Bonabeau, E., Dorigo, M., Theraulaz, G.: Swarm Intelligence: From Natural to Artificial Systems. Oxford University Press, New York (1999)
2. Bonabeau, E.: Swarm Intelligence. In: O'Reilly Emerging Technology Conference (2003)

3. Eberhart, R., Kennedy, J.: A New Optimizer Using Particle Swarm Theory. In: Proceedings of the Sixth International Symposium on Micro machine Human Science, pp. 39–43. IEEE Press, New York (1995)
4. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of 1995 IEEE International Conf. on Neural Networks, pp. 1942–1948. IEEE Press, New York (1995)
5. Hu, J., Wang, Z., Qiao, S., Gan, J.C.: The Fitness Evaluation Strategy in Particle Swarm Optimization. Applied Mathematics and Computation 217, 8655–8670 (2011)
6. Colorni, A., Dorigo, M., Maniezzo, V.: Distributed Optimization by Ant Colonies. In: Varela, F., Bourgine, P. (eds.) First Eur. Conference Artificial Life, pp. 134–142 (1991)
7. Dorigo, M., Maniezzo, V., Colorni, A.: Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man, and Cybernetics-—Part B: Cybernetics 26, 29–41 (1996)
8. Dorigo, J.M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation 1, 53–66 (1997)
9. Chu, S.C., Roddick, J.F., Pan, J.S.: Ant Colony System with Communication Strategies. Information Sciences 167, 63–76 (2004)
10. Karaboga, D.: An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report-TR06, Erciyes University, Computer Engineering Department (2005)
11. Passino, K.M.: Biomimicry of Bacterial Foraging for Distributed Optimization and Control. IEEE Control Systems Magazine 22, 52–67 (2002)
12. Chu, S.C., Tsai, P.W., Pan, J.S.: Cat Swarm Optimization. In: Yang, Q., Webb, G. (eds.) PRICAI 2006. LNCS (LNAI), vol. 4099, pp. 854–858. Springer, Heidelberg (2006)
13. Chu, S.C., Tsai, P.W.: Computational Intelligence Based on the Behavior of Cats. International Journal of Innovative Computing, Information and Control 3, 163–173 (2007)
14. Bishop, J.M.: Stochastic Searching Networks. In: Proc. 1st IEE Conf. on Artificial Neural Networks, London, pp. 329–331 (1989)
15. Chang, J.F., Chu, S.C., Roddick, J.F., Pan, J.S.: A Parallel Particle Swarm Optimization Algorithm with Communication Strategies. Journal of Information Science and Engineering 21, 809–818 (2005)
16. Tsai, P.W., Luo, R., Pan, S.T., Pan, J.S., Liao, B.Y.: Artificial Bee Colony with Forward-communication Strategy. ICIC Express Letters 4, 1–6 (2010)